Excel Developer Tip

Excel page

Tip archives

Working With Names in VBA

Most Excel users realize the value of using names in the workbook. This tip describes some VBA techniques that are useful when working with names.

- For more about names, refer to the **Naming Techniques** tip.
- You can **download the J-Walk Name Lister**, an Excel add-in that uses the techniques described in this tip to list various categories of names.
- For an enhanced version (as well as several other auditing tools), check out **Power Utility Pak**.

Listing all names

You probably know that you can get a list of all names by using the Insert Name Paste command, and then clicking the Paste List button. The list that gets generated omits hidden names and sheet level names. The subroutine below creates a list of *all* names and their references, beginning in cell A1.

```
Sub ListAllNames()
    Row = 1
    For Each n In ActiveWorkbook.Names
        Cells(Row, 1) = n.Name
        Cells(Row, 2) = " " & n.RefersTo
        Row = Row + 1
    Next n
End Sub
```

This subroutine works by looping through the Names collection.

Hidden names

Hidden names are names that are not visibile to the end user. Developers often use hidden names to store various types of information. To demonstrate, just save a range in HTML format using the Internet Assistant Wizard add-in (use the File Save as HTML command). After doing so, execute the ListAllnames subroutine and you'll find that your workbook contains more than a dozen new hidden names that contain the parameters you specified in the Internet Assistant Wizard. The Internet Assistant Wizard uses this information as default values the next time it is run.

A hidden name has its visible property set to False. The only way change this propery is with VBA. For example, the statement below makes the name MyRange a hidden name:

```
ActiveWorkbook.Names("MyRange").Visible = False
```

Sheet level names

A sheet level name is a name that is valid for a particular worksheet. To create a sheet level name, precede the name with the worksheet name and an exclamation point.

To find out if a particular name is a sheet level name, simply determine if the name's Name property contains an exclamation point. You can use the Like operator to do this. The expression below is True if MyRange is a sheet level

name.

```
ActiveWorkbook.Names("MyRange").Name  Like "*!*"
```

Linked names

A name can also refer to a different workbook. I call this type of name a linked name. To find out if a particular name is a linked name, simply determine if the name's RefersTo property contains a right bracket. The expression below is True if MyRange is a linked name.

```
ActiveWorkbook.Names("MyRange").RefersTo  Like "*[[]*"
```

> **NOTE**:
> A bracket is a special character when using the Like command, so it must be enclosed in brackets.

Erroneous names

It's not uncommon for a name to refer to a range that no longer exist (deleting a sheet can cause this). These types of names are responsible for the "phantom link" phenomenon in which Excel asks you to update links when you open a workbook -- even if no links really exist.

To find out if a particular name is an erroneous name, simply determine if the name's RefersTo property conains "REF!". The expression below is True if MyRange is an erroneous name.

```
ActiveWorkbook.Names("MyRange").RefersTo Like "*REF!*"
```